Executive Summary

1. Assignment Overview -

   The goal of this project is to use the TCP and UDP protocols to develop and implement a simple client-server application. Processing commands like PUT, GET, and DELETE, as well as key-value data storage, are the responsibility of the server. When a client requests data, the server needs to be able to store, retrieve, and delete it. The assignment also seeks to develop a grasp of network communication, data management, and debugging techniques by requiring the server to log a variety of operations, such as client connections and command processing. The project offers a chance to implement hands-on socket programming and learn the distinctions between the TCP and UDP protocols. The project's scope is limited to a single thread server which does not need to serve multiple clients at once.

2. Technical Expression -

   Completing this assignment gave me invaluable practical experience with client-server architecture and TCP socket programming. The requirement to prepopulate the server with data from a script file increased my grasp of file management and command parsing in Java. It was necessary to pay close attention to input validation and error handling when writing the logic to understand various commands, such as PUT, GET, and DELETE. I had to think critically about how to provide clear feedback to clients while maintaining a user-friendly interface.

   Although logging created an additional level of complexity, it was essential for monitoring server performance and identifying problems. It made me understand how important thorough logging procedures are to software development since they make performance monitoring and troubleshooting simpler.

   Overall, this assignment highlighted best practices in software development and enhanced my understanding of networking fundamentals and Socket programming using Java. I believe that increasing the project's modularity will present potential for improvement in the future. As an example, I want to minimize code duplication by creating a single application that can run over both TCP and UDP protocols—the protocol to use can be chosen using command-line options. Future revisions would be easier to maintain and scale thanks to the

codebase's increased flexibility and streamlining.

3. Use Case -

Cache Application

In a cache application, the TCP server can act as an in-memory data store to temporarily hold frequently accessed data, reducing the need for repeated database queries. This results in lower latency and faster data retrieval. Like Redis Cache which can store a key and value, and retrieve data using the key rapidly, this application can aslo be used similarly.

Session Management

When a user interacts with a web application, the TCP server can securely store user session data, including authentication tokens and user preferences, for the purpose of session management. The value may store the token value, and the authentication token string may serve as the key.